6 Jan 66

To: Information Technology Division
From: J. G. Cole, N. E. Lambert
Subject: Programmer's Summary
Date: 14 December 1965

This memo is a handy reference summary of the hospital computer. It contains parts of older, lengthy memos about the hardware, the iot's, the programming system, etc. Its purpose is to minimize time spent finding the correct memo.

Parts of this memo will be reissued periodically as the system evolves. Suggestions and improvements will be welcomed.

J. G. Cole
Nancy E. Lambert

JGC:NEL:zdg

CB' 81/11

[illegible]
[illegible] Digital Equipment Corp., 1962 for a detailed explanation of the instructions.

BASIC INSTRUCTIONS

add Y	40	Add C(Y) to C(AC)
and Y	02	Logical AND of C(Y) with C(AC)
cal	16	Equals jda 100.
dac Y	24	Deposit C(AC) in Y
dap Y	26	Deposit contents of address part of AC in Y
dch Y	14	Deposit C(AC_{0-5}) according to byte pointer in Y and rotate AC left 6 bits. If defer bit set, index the byte pointer in Y before depositing character.
dio Y	32	Deposit C(IO) in Y
dip Y	30	Deposit contents of instruction part of AC in Y
div Y	56	Divide C(AC,IO) by C(Y). Quotient in AC, remainder in IO. The next instruction is skipped unless the result overflows the AC.
dzm Y	34	Make C(Y) zero
idx Y	44	Index (add one to) C(Y). Leave in Y and AC.
ior Y	04	Inclusive OR of C(Y) with C(AC)
iot	72	See In-Out Transfer Group
isp Y	46	Index and skip if result is positive; see idx
jda Y	17	Equals dac Y plus jsp Y + 1
jmp Y	60	Take next instruction from Y
jsp Y	62	Jump to Y and save extended PC in AC.
lac Y	20	Load AC with C(Y)
law N	70	Load AC with the number N. If the defer bit is set, the resulting op code (71) will load AC with -N.
lch Y	12	Load AC_{0-5} according to byte pointer in Y and clear AC_{6-17}. If defer bit set, index the byte pointer in Y before loading character.
lio Y	22	Load IO with C(Y)
mul Y	54	Multiply C(AC) by C(Y). Result in AC, IO; sign in AC_0 and IO_{17}.
opr	76	See Operate Group
sad Y	50	Skip 1 instruction if C(AC) differs from C(Y)
sas Y	52	Skip 1 instruction if C(AC) is same as C(Y)
shift	66	See Shift Group
skp	64	See Skip Group
spo	74	See Special Operate Group
sub Y	42	Subtract C(Y) from C(AC)
tad Y	36	Two's complement add C(Y) to C(AC)
xct Y	10	Perform instruction in Y
xor Y	06	Exclusive OR of C(AC) with C(Y)

OPERATE GROUP

Two or more instructions may be combined by "or-ing" their addresses. The order in which the instruction is executed is:

1. cli cla
2. clf stf lap lat
3. cmi cma hlt
4. lai lia swp

cla	760200	Clear AC
clc	761200	Clear and complement AC (cla∨cma)
clf	76000f	Clear selected program flag (f=flag; 7=all)
cli	764000	Clear IO
cma	761000	Complement AC
cmi	770000	Complement IO
hlt	760400	Halt
lai	760040	Load AC from IO
lap	760300	Load AC from extended PC
lat	762200	Load AC from test word switches
lia	760020	Load IO from AC
nop	760000	No operation
stf	76001f	Set selected program flag (f=flag; 7=all)
swp	760060	Swap AC and IO (lai∨lia)

SPECIAL OPERATE GROUP

Two or more instructions may be combined by "or-ing" their addresses. The order in which the instruction is executed is:

1. cll sci scf szl
2. iif ifi scm
3. cml ida
4. idc
5. aai iai xai

aai	740003	Add AC and IO (-0 not changed to 0)
cll	740010	Clear link
cml	740004	Complement link
iai	740002	Inclusive OR to AC from IO
ida	740400	Index AC (-0 not changed to 0)
idc	741000	Index byte pointer in AC
ifi	742000	Inclusive OR to flags from IO (link,ring)
iif	744000	Inclusive OR to IO from flags (link,ring)
lfi	742050	Load flags from IO (scf∨cll∨ifi)
lif	744100	Load IO from flags (sci∨iif)
scf	740040	Special clear flags (ring,not link)
sci	740100	Special clear IO
scm	740200	Special complement AC (-0 not changed to 0)
szl	740020	Skip on ZERO link
xai	740001	Exclusive OR to AC from IO

SHIFT - ROTATE GROUP

Shift is an arithmetic operation. The sign bit is left unchanged and vacated bits are filled with the sign. Rotate is a logical operation and cycles the bits (including sign) in a closed ring. The number of steps is the number of ONE's in bits 9-17 of the instruction (9 max).

ral	661	Rotate AC left
rar	671	Rotate AC right
rcl	663	Rotate combined AC and IO left
rcr	673	Rotate combined AC and IO right
ril	662	Rotate IO left
rir	672	Rotate IO right
sal	665	Shift AC left
sar	675	Shift AC right
scl	667	Shift combined AC and IO left
scr	677	Shift combined AC and IO right
sil	666	Shift IO left
sir	676	Shift IO right

SKIP GROUP

Two or more instructions may be combined by "or-ing" their addresses. The intent of any single or combined skip group instructions may be reversed by making bit 5 equal to 1.

clo	651600	Clear overflow (spa∨sma∨szo∨i)
sma	640400	Skip on minus AC
sni	644000	Skip on NON-ZERO IO
spa	640200	Skip on plus AC
spi	642000	Skip on plus IO
spq	650500	Skip on plus quantity AC (sza∨sma∨i)
sza	640100	Skip on ZERO AC
szf	6400Of	Skip on ZERO flag (f=flag;7=all flags)
szm	640500	Skip on ZERO or minus AC (sza∨sma)
szo	641000	Skip on ZERO overflow (and clear overflow. Only add or sub set it)
szs	6400s0	Skip on ZERO sense switch (s=switch;7=all)

[illegible] [illegible] [illegible] 16 [illegible]

[illegible]	[illegible]	[illegible]
bedtr,	iot 6216	bedspace transfer. (see a;p10)
bgi,	iot 4600	initialize item buffer for put (see e)
clrdr,	iot 240	clear reader buffer, one return
crf,	000363	flag used by edit, c(crf)=0,cr and lf changed to space
dcdtd,	iot 5360	decode time and date (see d)
ddtmod,	iot 360	change ddt mode. one return, restart mode (p+3, b3) set same as ac, b3.
dnlv,	iot 4700	jbh iot for question no. control (see c)
dnm,	iot 4200	decode number (see f)
dnm2,	iot 4240	double precision dnm
ecm,	iot 220	enter control mode. one return, set tt to use iddt's set of terminators.
edit,	iot 4160	removes backslashes and rubouts from a text string
eebm,	iot 2340	enter 8-bit mode. store all type-in chars. on tt in 65(ttno) as 8-bit until all ones char. received
erim,	iot 5540	enter simulated readin mode
exara,	iot 6203	expunge addressed rewritable item, apr
exarl,	iot 6315	library
exarm,	iot 6255	message
exarp,	iot 6335	programmer's
exarr,	iot 6355	research
exia,	iot 6205	expunge an item, apr
exil,	iot 6305	library
exim,	iot 6245	message
exip,	iot 6325	programmer's
exir,	iot 6345	research
exp,	iot 6215	expunge patient parameters. (see a;p10)
extl,	iot 6310	expunge a toc'd item, library
extm,	iot 6250	message
extp,	iot 6330	programmer's
extr,	iot 6350	research
fail,	iot 5060	jbh iot, types fix and causes restart (c - p9)
fac,	000047	ac for floating point routines
fadd,	iot 4340	floating add, 1 argument
fdac,	iot 4460	" dac, " "
fdiv,	iot 4420	" div, " "
fix,	iot 4520	" to fixed format
fix2,	iot 4560	double precision fix
flac,	iot 4440	floating lac, 1 arg
float,	iot 4500	fixed to floating format

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	floating [illegible]
fov,	000422	trap location for floating point routines on overflow and underflow conditions
fov+1,	000423	on trap to fov, contains a jmp to instruction following the call that caused the trap
fsa,	000120	text pointer
gdtime,	iot 2360	get down time
get,	iot 4620	(see e)
getlp,	iot 2260	get list pairs. r1 (used by evd, tot)
gtd,	iot 2060	get time and date. r1, ac(days since 1/1/1849) io(minutes since midnight)
gty,	iot 2320	get teletype. c(65) no. of tt wanted. two returns
hltbad,	iot 5460	types "-x-" and traps to location 77
hltgud,	iot 5440	types "-thank you-" and executes a hlt
holdpf,	iot 2520	associates the number of your drum field to the specified pending file if available
icm,	iot 200	in control mode status. one return. c(ac)=0 if not, c(ac)=-0 if tt in control mode.
jans,	000364	jbh pointer to text of last response
jbf,	000373	jbh register. 1 - brief mode, 2 - long mode
jlist,	000400	jbh register. values <0 - simulating 0 - normal, 1 - error correction, 2 - listing
jntp,	000371	jbh next text pointer - points to next free byte in jbh buffer
jorg,	000365	jbh pointer to first instruction after last init executed
jqn,	000376	jbh six digits of current question
jqn+1,	000377	no. in internal
jqnp,	000372	jbh pointer to next free location in question number table
jtis,	iot 5560	jbh "tis"
jtp,	000370	jbh text pointer, moves along buffer
jvan,	000415	jbh pointer to answer of validating question. also flag as to whether validating has happeded.
lcm,	iot 260	leave control mode, one return. (see ecm)
lebm,	iot 2500	leave 8-bit mode
mbbcd,	iot 6140	backspace 1 block, bcd
mbbin,	iot 6120	backspace 1 block, bin
mrbcd,	iot 6000	read bcd record, 84. characters
mrbin,	iot 6060	read binary 200. word record
mrew,	iot 6160	rewind
mwbcd,	iot 6020	write bcd record, 84. characters
mwbin,	iot 6040	write binary 200. word record
mweof,	iot 6100	write end of file mark

```
opb,     000[illegible]    tty output buffer

[illegible]
pua,     [illegible]       punch paper tape, [illegible]
                             character in io, b10-b17, one return
pus,     iot [illegible]   (see e)


rabi,    iot 6261      read addressed block from index
rabl,    iot 6301        library
rabm,    iot 6241        message
rabp,    iot 6321        programmer's
rabr,    iot 6341        research
rara,    iot 6213      read addressed rewritable item, apr
rarl,    iot 6313        library
rarm,    iot 6253        message
rarp,    iot 6333        programmer's
rarr,    iot 6353        research
rck,     iot 2300      read millesecond clock
reask,   iot 5100      jbh iot, identical to fail except no typeout
rebl,    iot 6303      read and expunge addressed block, library
rebm,    iot 6243        message
rebp,    iot 6323        programmer's
rebr,    iot 6343        research
restar,  iot 5240      jbh iot - restart in simulation mode from
                         last init
rita,    iot 6206      read an item, apr
ritl,    iot 6306        library
ritm,    iot 6246        message
ritp,    iot 6326        programmer's
ritr,    iot 6346        research
rpa,     iot 20        read paper tape, alpha
                         character into io, b10-b17, one return
rpb,     iot 5520      read paper tape - binary
rpp,     iot 300       read p pointer. ac (1-37) return 1 io(p-p+36)
rppb,    iot 6207      read patient parameters by bedspace.
                         ac(core addr), b.s. in 42 - reads unit no.
                         into 40,41, reads 34. parameters
rppu,    iot 6210      read patient parameters by unit no.
                         complement of rppb.
rtil,    iot 6311      read a tee'd item, library
rtim,    iot 6251        message
rtip,    iot 6331        programmer's
rtir,    iot 6351        research
rty,     iot 2240      release teletype
```

[illegible]	[illegible]	library
[illegible]	[illegible]	message
[illegible]	[illegible]	programmer's
rwabr,	iot 635[illegible]	research
sickrt,	iot 5500	teletype error recovery routine
snm,	iot 4220	set up number (see f)
snm2,	iot 4260	double precision snm
sts,	000121	storing text pointer
stv,	iot 4020	call stv (see h)
stvl,	iot 4060	return to stv in lose mode
stvw,	iot 4040	return to stv in win mode
sua,	000122	stv interpretive pc
sug,	000424	stv - output pointer
suq,	000431	stv resets output pointer to the contents of this location. (see h)
sus,	000411	stv ignore register, non-general
sutd,	iot 4140	set up time and date (see d)
tas,	iot 2540	type a string. like tos, but with terminator stored in io, b0-b5. return r2 with ac pointing at end of string.
td1to2,	iot 5400	time/date, convert 1 to 2 word format (see d)
td2to1,	iot 5420	time/date, convert 2 to 1 word format (see d)
tis,	iot 2220	type in string. ac points to input buffer.
tos,	iot 2160	type out string. ac points to beginning of string. eom terminates string.
toss,	iot 2020	type out string, special. tos for tot
ttcks,	iot 2140	check teletype status
tth,	iot 2000	tt hang. for tot
tyi,	iot 100	type in character to ac, b0-b5, low order bits zero, two returns, r2 sucessful
tyihng,	iot 2200	teletype error recovery
tyin,	iot 5020	jbh iot to accept a typed in answer (see c)
tyo,	iot 60	type a character from ac, b0-b5. transparent, two returns, r2 sucessful.
uplv,	iot 4720	jbh iot up level, question no. control (see [illegible]
wabi,	iot 6260	write addressed block on index
wabl,	iot 6300	library
wabm,	iot 6240	message
wabp,	iot 6320	programmer's
wabr,	iot 6340	research
waia,	iot 6217	write addressed item on apr

```
wita,     iot 6204    write an item, apr
witl,     iot 6304      library
witm,     iot 6244      message
witp,     iot 6324      programmer's
witr,     iot 6344      research
wnbl,     iot 6302    write non-addressed block, library
wnbm,     iot 6242      message
wnbp,     iot 6322      programmer's
wnbr,     iot 6342      research
wnra,     iot 6212    write non-addressed rewritable item, apr
wnrl,     iot 6312      library
wnrm,     iot 6252      message
wnrp,     iot 6332      programmer's
wnrr,     iot 6352      research
wpid,     iot 6211    write patient identifier. (a - p9)
wpp,      iot 320     write p pointer. ac (1-37) io (new p ptr)
wtil,     iot 6307    write a toc'd item, library
wtim,     iot 6247      message
wtip,     iot 6327      programmer's
wtir,     iot 6347      research

xdnlv,    iot 4740    jbh iot - index and down a level (see c, p9)
```

bibliography

a - big hospital iots, 12/8/64
b - revised teletype iot's, 12/22/64
c - the new job hunter, 6/17/64
d - time and date, 10/16/64
e - put and get, 11/2/64
f - jbh, stv, and common routines, 6/2/64
g - exec's pooled list storage, 10/27/64
h - common routines, part 2, 11/19/64

[illegible]

[illegible]	[illegible]	[illegible]
1	p	ac
2	p+1	pc
3	p+2	io
4	p+3	flags; b0=xddt, b1=iddt, b2=dzm pc, b3=ddtmod
5	p+4	b16-b17 = seg running
6	p+5	tt calling; b0 set if control type tt
7	p+6	time left on cue
10	p+7	seg 0 ptr
11	p+10	seg 1 ptr
12	p+11	seg 2 ptr
13	p+12	seg 3 ptr
14	p+13	a for x16
15	p+14	b "
16	p+15	c "
17	p+16	d "
20	p+17	e "
21	p+20	f "
22	p+21	g "
23	p+22	h "
24	p+23	programmer's index ptr
25	p+24	program number
26	p+25	program name
27	p+26	"
30	p+27	"
31	p+30	typist's initials
32	p+31	date
33	p+32	time
34	p+33	
35	p+34	
36	p+35	number of characters handled
37	p+36	number of tt's held
40		unit number (high order)
41		unit number (low order)
42		bedspace
43		iop max
44		tis max
45		error code
46		
47		fac; also unit number 2 (high order)
50		fac; also unit number 2 (low order)
51		bedspace 2
52-55		
56		char count
57		line count
60-62		(used by iddt)
63-64		
65		current scanner line
66		teletype name number
67		crash trap buffer
70		crash ac
71		crash pc
72		crash io
73		x16 userac
74		x16 userpc
75		x16 userio
76		
77		hitbad return

[illegible]

tt errors (word 45)

400000	bit 0	tt not yours
200000	bit 1	tis max exceeded
100000	bit 2	tos ran over end of core
2000	bit 7	14 echo checks
1000	bit 8	interrupted by null
400	bit 9	tt did not turn on (gty)
200	bit 10	eot echo check
4	bit 15	tt is off
1	bit 17	line open

iop errors (word 45)

1	illegal drum address
2	no more blocks left in area
3	(rewrite) version number disagrees
4	iopmax < starting address on read
5	length is zero or greater than 10000
6	(big hardware trouble)
7	attempt to write on another programmer's quarter track
10	(rppu) no such unit number
11	no such nursing station or bed number
12	(write item) unit number disagrees with 40 and 41
13	(wita) illegal apr item class (like 0)
14	ordinary data error - try again
15	(mag tape) read eof
16	(mag tape) read end-of-tape mark
17	(big tape controller error)
20	apr rewrite error (e.g., first or second words disagree)
21	attemping to expunge with wrong drum address

PROGRAMMING SYSTEM

1) Programmer's Index (p#23)

```
address of previous block
word count this block
                type
               /name
                name
8 words        | version
per file      <  time,date
               | -
               | address
               \[address macro table]
```

```
type:       0 binary
            1 -
            2 core image
            3 english
            4 symbols
```

2) Handle (32335,sa100)

programmer's index commands

```
add name,ver,type     drum address[es]
b    (list with addresses)
delete name,ver,type
find name,ver,type
edit    (ecm or Kecm)
kill
list
```

mag tape commands (tape rewinding at end)

```
seduce   unit?
print   unit?
read   unit?   block?
write   unit?   name?
org   unit?
```

[illegible]	nam	to pass 1
[illegible]	nam,ver	to pass 2
C	nam.ver	continue current pass
J		jump block at end
B	nam,ver	file binary in index
S	nam,ver	file symbols in index
A		alphabetic symbol print
P		symbol punch on paper tape
H		halt
T	nam.ver	load binary symbols
I		initialize symbols
E	num	extra features bit 14,define usw's=0 bit 15,lgi typeout bit 16,print all

4) Editor (450,sa100)

Restart at 100 to save buffer, 101 to kill buffer

"A"	LOC"A"	Append
LOC"C"	LOC;LOC"C"	Change
LOC"K"	LOC;LOC"K"	Delete
"P"	LOC;LOC"P"	Print
S"F",	U"F",NAME"N",VER"V"	File in index, append from index
1"H"		Halt
"R"	LOC"R"	Append internal paper tape
"Y"	LOC"Y"	Append concise paper tape
"T"	LOC;LOC"T"	Punch internal paper tape
NUM"Z"		Tape feed paper tape
"W"		Verify internal paper tape
"X"		Seg to lister
R"Z"		Punches in lines of tape feed
LOC/		Open line
RUBOUT		Delete open line
.		Value of current line
↑		Open previous line
RETURN		Open next line
@		Value of last line

In text mode, backslash and rubout have normal meaning to terminate text input type "carriage return, alt mode."

5) [illegible]

E	Jag to editor
R RE	Append internal paper tape
Y YE	Append concise paper tape
FNUMBER	Append saved file
W "W" W2,7,6-@	Write (print) pages
K K3,9,14-17	Delete pages
HALT ←—	Halt
@	Number of pages
C50	Wait at beginning of page
B10	Ring bell
S72	Width
L2	Space between lines
T10,14,12	Tabs
I67,67,66	Page length
M60	Lines/page
#L #P #L,P #	Number pages, number lines
P "P"	Proceed
O	Octal
D	Decimal

6) NDDT

CONTROL CHARACTERS:

	"B"	Flush all breakpoints
ARG;ARG	"B"	Set breakpoint
ARG	"C"	Change cores
	"D"	Turn off TT and hang
ARG	"E"	Effective address search (M#)
	"F"	See below
ARG	"G"	Set PC, go
1	"H"	Halt
	"K"	Kill all but initial symbols
	"L"	Punch, see below
ARG	"N"	Not word search (M#)
	"O"	Type "Q" as time and date
ARG	"P"	Multiple proceeds
	"p"	Proceed
	"Q"	Value of last quantity typed
ARG	"R"	Set radix of numeric typeout
	"S"	Type as sqoze
1	"T"	Clear buffer; read symbol tape (C#)
	"T"	Read symbol table image from drum (F#) (C#)
	"U"	Logical OR
	"V"	Verify core against binary tape image ((M#,F#)
1	"v"	Clear buffer; verify core against paper tape
ARG	"W"	Word search
ARG	"X"	Execute ARG
1	"Y"	Clear buffer; yank binary tape (M#)
	"y"	Read binary tape image from drum (M#,F#)
ARG;ARG	"Z"	Zero core (M#+3)

FILE HANDLING:

A	"F"	Add file (F#-2 to F#+2)
C	"F"	Get startup program
D	"F"	Delete file (F#-2 to F#+2)
F	"F"	Find file (F#-2 to F#+2)
L	"F"	Unsave from library
M	"F"	Unsave core image from message tracks
S	"F"	Save core image on programmer's tracks
U	"F"	Unsave from programmer's tracks

PUNCHING:

1	"L"	Punch loader
-0	"L"	Punch loader; set to punch only non-zero core (M#+3)
ARG;ARG	%	Punch data
	*	Punch register
ARG	@	Punch jump block

TYPEOUT MODES:

A!	Numeric location
R!	Symbolic location
F!	Floating contents
C!	Numeric contents
I!	Internal code contents
S!	Symbolic contents

SPECIAL REGISTER EXAMINATION:

B#/	Location of breakpoint # 0
C#/	Which core image is being examined
E#/	Floating format word
T#/	Bottom of symbol table
G#/	Number which is generated all over core ('Z)
M#/	Mask, lower limit, upper limit
F#-2/	Mask for file operations
F#-1/	Type
F#/	Name, left — "ABC"
F#+1/	Name, right "␣␣D
F#+2/	Version
F#+3/	Time and date
F#+4/	Spare
F#+5/	Address ———— for core image.
F#+6/	Macro table address

[illegible]

ARG /	Open register
ARG ;	Open register, 16-bit address
ARG [(K)	Open register, force numeric typeout
ARG] (M)	Open register, force symbolic typeout
=	Numeric typeout of "Q"
←	Full symbolic typeout of "Q"
?	Internal code typeout of "Q"
ARG "	Take ARG as internal code
LINEFEED	Close register
RETURN	Close register, open next
ALT MODE	Close register, open register addressed by "Q"
>	Close register, open register addressed by "Q", same sequence
↑	Close register, open previous
&	Logical and
.	Current location, take number as decimal, symbol constituent
'	Open in type in mode
$	Print as floating point

Error	Cause	Effect
	Undefined Symbols	
usα	Undefined symbol (α indicates where found):	set symbol=0
c	in a constant	
d	in size of dimension array	
f	in offset count	
i	in argument of Ø1f or 1if	
l	in a location assignment	
m	in storage word generated by a macro call	
o	in argument of equals or opsyn	
p	in a parameter assignment	
r	in the count of a repeat	
s	in the argument of a start	
t	in a multi-syllabic address tag	
w	in a storage word	
uwd	undefined symbol in argument of a pseudo-instruction	

Error	Cause	Effect
	Relocation Errors	
irα	Illegal relocation; see undefined symbol section for meaning of α	Relocation

Error	Cause	Effect
	Multiple Definitions	
mdt	Multiply defined tag	no redefinition
mdv	Multiply defined variable	no redefinition
mdd	Multiply defined dimension	no redefinition

[illegible]

mnd	Macro name disagrees. (define-terminate)	First name used
ich	Illegal character	The character is ignored
ilf	Illegal format	Characters ignored until next tab or carriage return.
ipa	Improper parameter assignment.	The assignment is ignored.
vld	Variables location disagrees on different passes.	Condition ignored.
lgi	Location gone indefinite	If bit 15 set, this error message printed on Pass 1.
irx	Illegal operation with relocatable symbols.	
pnt	Not an error. Result of print pseudo-instruction.	

In the event of the following error conditions, assembly cannot continue.

cld	Constants location disagrees on different passes.
tmc	Too many constants
tmp	Too many parameters. (macros)
tmv	Too many variables.
sce	Storage capacity exceeded
ise	Call ALICE.

A		61
B		62
C		63
D		64
E		65
F		66
G		67
H		70
I		71
J		41
K		42
L		43
M		44
N		45
O		46
P		47
Q		50
R		51
S		22
T		23
U		24
V		25
W		26
X		27
Y		30
Z		31
0		12
1		01
2		02
3		03
4		04
5		05
6		06
7		07
8		10
9		11
[&]	+	60
-		40
/		21
[#]	=	13
.		33
$		53
°		73
[@]	'	14
[%]	(	34
*		54
[□]	)	74
blank		00

TELETYPE AND INTERNAL CODES

INTERNAL	ASCII	CHARACTER
00	040	SPACE
01	041	!
02	042	"
03	043	#
04	044	$
05	045	%
06	046	&
07	047	'
10	050	(
11	051	)
12	052	*
13	053	+
14	054	,
15	055	-
16	056	.
17	057	/
20	060	0 (ZERO)
21	061	1
22	062	2
23	063	3
24	064	4
25	065	5
26	066	6
27	067	7
30	070	8
31	071	9
32	072	:
33	073	;
34	074	<
35	075	=
36	076	>
37	077	?

40	100	ˇ
41	101	A
42	102	B
43	103	C
44	104	D
45	105	E
46	106	F
47	107	G
50	110	H
51	111	I
52	112	J
53	113	K
54	114	L
55	115	M
56	116	N
57	117	O
60	120	P
61	121	Q
62	122	R
63	123	S
64	124	T
65	125	U
66	126	V
67	127	W
70	130	X
71	131	Y
72	132	Z
73	133	[
74	033	EOM
75	135	]
76	136	↑
77		WARNING

7700	000	NULL OR BREAK OR "@"
7701	001	"A"
7702	002	"B"
7703	003	"C"
7704	004	EOT
7705	005	"E" OR WRU
7706	006	"F" OR RU
7707	007	"G" OR BELL
7710	010	"H"
7711	011	TAB
7712	012	LINE FEED
7713	013	"K"
7714	014	"L" OR FORM FEED
7715	015	CARRIAGE RETURN
7716	016	"N"
7717	017	"O"
7720	020	"P"
7721	021	"Q"
7722	022	"R" OR TAPE
7723	023	"S" OR RDR OFF
7724	024	"T"
7725	025	"U"
7726	026	"V"
7727	027	"W"
7730	030	"X"
7731	031	"Y"
7732	177	"Z"
7733	022	"["
7734	034	BACKSLASH
7735	035	"]"
7736	036	"↑"
7737	037	"←"
7740-7775	UNUSED	
7776	137	←
7777	177	RUBOUT

To: Information Technology Division
From: A. K. Hartley Mastinst. -926- [illegible]
Subject: Changes in Programming System for Exec III (Revised)
Date: 23 February 1966

Memo #555
Descriptors:
Exec III
User Programs
Programming System

FILES

All files saved by Editor and Midas are written on free information on the middle third of the drum. Free information does not vanish when you halt. Temporary buffers and DDT saved core images are written on held quarter-tracks and are not saved through halts. In order to save a core image forever, one may add it to an index using Handle and it will then be copied from held to free automatically. It is illegal to file anything with Editor or Midas without first obtaining an index with Handle.

core

S"F" in DDT
call handle: Add, Name, Version, c
→ if asks for a drum address -- type in.

There are a few changes in file formats, which should make no difference to the user but are hereby recorded for those who may be interested. Symbol tables are now filed as type 5, and are stored in the same format as binary paper tape images; toc'd items with the length of the toc in its first word. Core images are filed as standard items of length 7742, with an OWNWD of 44032.

EDITOR

1. "R" or arg"R" - reads new internal paper tape.
2. "W" or arg"W" - reads old internal paper tape.
3. "V" or arg;arg"V"- verifies new internal paper tape.
4. name,vers"S" - files the current state of the text buffer as the English file name,vers on the Fastrand and enters it into the programmers index.

5. name,vers"U" or arg;name,vers"U" - appends the English file name,vers from the Fastrand.

6. loc1;loc2= - gives number of lines in octal between loc1 and loc2 (inclusive). As always, loc= gives the number of lines from the beginning of the buffer through loc.

7. The updating of ".", the current location, has changed slightly.

   a. When anything is appended, whether from the Fastrand, reader, or Teletype, "." is set to the first line of the appended text.

   b. When anything is changed, "." is set to the first line of the new text.

   c. When anything is deleted (or changed to nothing), "." is set to the line before the deleted text. If there is no line before the deleted text (the first line of the buffer was deleted), "." is set to the line after the deleted text.

8. 1"G" or 101"G" kills the entire buffer.

MIDAS

1. The control language now uses "tis" and "edit." This means that backslash deletes the last character, rubout deletes the whole input string, and alt. mode terminates the command. In the input string spaces are ignored and 12-bit characters are illegal.

2. If a symbol table is filed (S name,vers) at any time during pass 2, any binary output that has accumulated will be destroyed. Thus, when filing binary and symbols at the end of pass 2, binary must precede symbols.

DDT

1. Breakpoints:

The breakpoint facility has been expanded to include multiple proceeds through any of the four breakpoints, fast multiple proceeds in certain cases, and the ability to type the contents of a specified register each time a breakpoint is hit.

There are four special registers for each breakpoint, beginning at B# (or B0#), B1#, B2#, and B3#. The first register contains the location of the breakpoint. This is a 15-bit address with the core image of the breakpoint in the high order bits. The second register contains the proceed count (positive) for the breakpoint, or the number of times to go through this breakpoint before returning to DDT. The third register contains the address of a location to be typed. If bit 0 is a 0, the register will be typed only when the proceed count is counted down to 0 and control returns to DDT. If bit 0 is a one, the register is typed each time the breakpoint is hit, and if the proceed count is not yet 0, the users program continues to run. The fourth register in each group is unused but should not be changed from 0.

Breakpoints may be set by typing directly into Bn# (n=0,1,2,3) the desired address with the core image in the high order 6 bits. They may also be set by typing n;loc"B", to set breakpoint n, at loc, in the current core. If n>3, the next free breakpoint is assigned and Bn# is typed out. As before "B" clears all breakpoints but, in addition, n"B" (n=0,1,2,3) clears breakpoint n. Since breakpoints may not be set below location 40, the memory protect limit, n"B" can be used both to set and clear breakpoints depending on the value of n.

Proceed counts may be set by entering a number into Bn#+1 or by typing n"P" after breaking at the desired breakpoint. Note that multiple proceed after an illegal instruction or hitting the break-key is meaningless and, if num"P" is typed under these circumstances, num will be ignored.

The register to type is set by entering loc or add loc (bit 0=1) into Bn#+2.

Fast proceeds take place (in core 16) for breakpoints at any instruction except cal, jda, jsp, lap, and xct.

2. Core Images:

There are now 7 core images (excluding DDT core). Registers 0-36 are common to all core images. Since I: is equivalent to 0:, I: will not generate a core; however, I+40: or 1"C" will generate a core if there isn't one already.

The AC, PC, IO, and flags for each core are stored in registers 40, 41, 42, and 43.

3. Symbols:

The ability to define and kill individual symbols has been restored.

a. sym"K" - removes "sym" from a symbol table. As before, "K" kills all but the initial symbols.

b. sym, - defines sym locally (i.e. over the range of the current core) with the value of the current location.

c. sym< - defines sym locally with the value of the address part of the last quantity typed.

d. (sym) - defines sym globally (in the initial symbol table) with the value of the last quantity typed.

One may now read symbol tables from either the drum or paper tape.

a. arg"T" - reads symbols from the drum, using F# through F#+2 for name and version, and defines the symbols from the beginning of the current core to arg. "T" defines the symbols for all cores except DDT core.

b. -arg"T" reads paper tape symbols, defining them from the beginning of the current core to arg. -0"T"defines paper tape symbols for all cores.

Since the method of setting the lower limit of symbol table definition using arg: has caused much confusion it has been eliminated. The lower limit is now I or the bottom of the current core.

4. Save and unsave are faster. Save is on the held (temporary) area of the drum. Unsave is from either held (temporary) or free (permanent), depending on the drum address. M"F" and L"F" no longer exist since U"F" serves all purposes. When a core image is unsaved, nothing is changed in it, in case it is a crashed program that one wishes to examine. Consequently the Teletype number may not correspond to the current Teletype and may have to be changed before the program can be run.

5. A"F" and P"F" have been removed. Use Handle for these purposes.

6. Remember that if you wish to execute a Fastrand or tape iot, as iot mumble"X", and if bit 17 is a 0 and there is an error you will end up at IOFTSU which may have some bizarre effect if the possibility is not forseen

A. K. Hartley

AKH:dcr

7 Apr 66

```
1/U2/U3/U4/U
0/
3I@S Y5 VFI XF2GY3@C D@SXYEU !L!L&%"
"C2
[
223)    IOT 200




@








C"F"  C? EDITOR  S? Y

"W"

??0/
3I@S Y5 VFI XF2GY3@C D@SXYEU !L!L&%"
"C2
"W"
0/
3I@S Y5 VFI XF2GY3@C D@SXYEU !L!L&%"
"C2   ?@
706)    XCT 702
C"F"  C? EDITOR  S? Y

"Y"
0/
TRAC IV FOR HOSPITAL MACHINE 1-1-65

              LS=34          /BEGIN ERROR COMMENT
T4.1,1"S"     145530

223)    IOT 200
101"G"

"Y"
T4.1,2"S"     145765

223)    IOT 200
101"G"

"Y"
T4.1,3"S"     144705
P
223)    IOT 200
1"H"

8:35 PM,
8:38 PM 4/9 400-1
DDT CNM

C"F"   C? HANDLE   S?

36)    7742
C"F"   C? HANDLE   S? Y

HANDLE

NAME OF INDEX?   CNM1
←LIST
E    T4.1      1
E    T4.1      2
E    T4.1      3
←HALT
8:39 PM
316)    IOT I 2401
C"F"   C? EDITOR   S? Y

T4.1,2"U"
0/
TRAC IV 1-1-65 : PART 2

/DEFINITIONS

DEFINE        DP A
O A 1         D A
TERMIN

DEFINE        LT A
0;5"P"
TRAC IV 1-1-65 : PART 2

/DEFINITIONS

DEFINE        DP A
O A 1         D A
5;150"P"
O A 1         D A
TERMIN

DEFINE        LT A
F A 1         L A
TERMIN

DEFINE        AM B
A B           D B

223)    IOT 200
101"G"

T3;U#T4.1,3"U"
0;10"P"


/NUMBER USING FUNCTIONS

0BI,          C BR2
              N T3
BNW.          C BWR
```

NEW EXEC

5 April 66

Example of MIDAS assembly

```
C"F"   C? EDITOR   S? Y

MAG2,2"U"
30/

EDIT=IOT I 100
33=33
33/
HALT=IOT I 2401
"L"
0;34"C"
SYSMAG2,3 4-1-66 JGC
RESTPC=32
MRPRF=MRPR+40
MWPRF=MR\WPR+40


7/
IOPMAX/      0
14/
400/    172/

GO #
GO1 13/
              TOS
              LAI
              SNM 3    A+\\    SNM"U"13

../
SYSMAG2,3 4-1-66 JGC
GO1 15/
              SNM"U"13
GO7 2/
              JMP GO8
."A"
   DZM ERCODE

. 7/
              JSP ERR    ↑
              SAS (2000        SAS (402000

GO8"C"
GO8,    DZM ERCODE
   LAW MADBUF\\\\GBUF

. 5/
              LAC ERCODE
              SAS (2000        SAS (402000

. 1/
              JSP ERR
."A"
   DZM ERCODE

.-1/
              JSP ERR
              DZM ERCODE
              LAW MAGBUF
. 5/
              SAS (2000        SAS (402000
               JSP ERR
               DZM ERCODE
               LAW MAGBUF
. 5/
               SAS (2000      SAS (402000

TTERR 2"K"
TTERR 2/
               JMP .        DAP TTER2
TTER1,    LAW 2
   SAS ERCODE
   HLT
   LAW TXCR
   TOS"U"1
    JMP TTER1
TTER2,    JMP .

ERX-4/
               SNM 3        SNM"U"13

ERX/
ERX,           SNM 3    ERX,    SNM"U"13

TXERC 1/
"L"    #
./
TXERC,         TEXT .     ERCODE=#.

TEM,           0
MAG2,3"S"      120063

223)    IOT 200
C"F"   C? MIDAS   S? Y



#1 PTAPE3,1
PTAPE 3.01 WITHOUT MAP MACROS - PASS 1

#C MAG2,3
SYSMAG2,3 4-1-66 JGC - PASS 1

#2 MAG2,3
SYSMAG2,3 4-1-66 JGC - PASS 2

#J

#B MAG2,3

#S MAG2,3

#
```

save

TITLE

START

TITLE

START

ND[illegible]

MIDAS
EDITO
example

5 Apr 66

```
]\
P
P
/    W EF
INR"G"

A'A'

TCI)    TYI
1"H"



-X-






P
5:28 PM 4/5/1966   40(40)
CALL NDDT   CNM

"C"?
EDITUNEDITU
C"F"  C? EDITOR   S? Y

"A"
NOW IS THE TIME FOR ALL GOOD MEN
TO COME TO THE
PR\ARTY

"P"
NOW IS THE TIME FOR ALL GOOD MEN
TO COME TO THE
PARTY
"K"?@=2
0;2"K"
"P"?
"A"
TITLE
200/GO,    LAC
I\    DAC
START GO

"P"
TITLE
200/GO,        LAC
               DAC
START GO
THIS"N"
12"V"
```

```
156)   IOT 100
C"F"  C? MIDAS  S? Y

#1 THIS,12
?
#1 THIS,12
TH\ITLE - PASS 1

#2 THIS,12
TITLE - PASS 2

#J

#B THIS,12

#F#?
#S

#
2215)   IOT 100
F#/   0   THIS
UF#/   0   THIS
               U

"Y"U
200/   DZM 4175

201/   DZM 4172
C"F"  C? EDIT ILLEGAL NAME  C? EDITOR  S? Y



THIS"N"

12"V"
U"F"
"P"


TITLE
200/GO,     LAC
            DAC
START GO
@=3


2"C"
JMP
"P"
TITLE
200/GO,     LAC
JMP
   START GO
2"C"
   JMP
2"P"

            JMP
@=3
LOC/U2/
            JMP
0/
TITLE
```

```
1/
200/GO,     LAC    ↑
TITLE   ↑
?
200/GO,     LAC
            JMP
START GO
?
D"F"?C"F"?
@
156)   IOT 100
C"F"  C? MIDAS  S?

62)   0   ]\
C"F"  C? MIDAS  S? Y

#1 THIS,12
TITLE - PASS 1

#2 THIS,12
TITLE - PASS 2

#J
#B THIS,12

#S THIS,12

#1?
#
2215)   IOT 100
F#/   0   THISU
F#/   0   THI"
F# 1/   0   S"
63/   0
F# 2/   0   12
F# 1/   63   S"
"T"
I I I:   0
"Y"
200/   LAC
GO+1/   DAC
GO+2/   0
GO+3/   0
200"G"

GO+2)   0

R"F"?
Q    U
GO+3/   0
F# 5/   AND 8S+23
I!
F# 5/   "$2
F# /   THI
F# 1/     S
F# 2/     *   =   12
S!
200/   LAC
GO+1/   DAC   B#/    -0
F#-1/   0
F#-2/   4S   =   17   T#/   8S 76   =   475
```

```
C"E"  C? EDITOR  S? Y

"O"?
A\?
"A"
XY

"O"?
XY"O"U
"A"
THIS
"P"
"P"
XY
THIS
"P"
"K"?
"K"?@=2
0;2"K"
"P"?
P
?
1"H"


-X-









6S)    HLT
VJ\
/OCE
1"H"


-THANK YOU-








\
6:09 PM 4/5/1966   40(40)
CALL


-X-
```

```
C"F"  C? MIDAS  S? Y
\
```

Andy Munster

OLD EXEC

```
 \4/196
2
2:03 PM 4/4/1966  14(14)   RR1
CALL EDITOR  CNM

"A"
THIS IS THE
TEST OF THE
EDITOR

"P"
THIS IS THE
TEST OF THE
EDITOR
3"A"U
???2"A"
NOW
IS
THE
TIME

"P"
THIS IS THE
TEST OF THE
EDITOR
NOW
IS
THE
TIME
S;7"P"U
 ?3;5"P"
```

```
C"F"  C? MIDAS  S? Y
\

#1 FOO,11
TITLE - PASS 1

#2 FOO,11
TITLE - PASS 2

#J
#B F]\
2215)    IOT 100    "P"
?
#B FOO,11
2215)    IOT 100    "P"
?
#B FOO,11

#S FOO,11

#]
\
\2

221\\\15\5/    IOT 100

F#A"?\0\? FOO"
\
\\\\]\\\
\\\\2216\6/\    \

F#/    0    FOO"
F#+1/    0
F# 2/    0    11
"Y"

200/    LAC
201/    DAC

"T"

GO/    LAC

1"H"
```

[illegible]

CR LF

Can get several files in by
#C Name, Version

continue on pass 2 with new chunk.

```
IS
THE
@"P"
TIME
@=6
@=6
1?
\?1"H"


-X-
```

call a program under NDDT.

(can call TRAC this way)

```
10:18 AM 4/5/1966  14(14)  RR1
CALL NDDT  JGC

C"F"  C? EDITOR  S? Y
```

start

yes

~~156)   IOT 100\\0~~
~~A#~~
~~"P"~~

```
"A"
TITLE
200/GO,   LAC
   DAC
START GO
```

Ⓐ ⒸⓇ ⓁⒻ Ⓐ

Midas needs a title & "start" at end.

```
"P"
TITLE
200/GO,      LAC
             DAC
START GO
FOO"N"
11"V"
S"F"   4415

156)   IOT 100
```

print.

~~FOO,11"S"~~ Naming in new exec.

Save.

Ⓑ

Naming in NDDT for progs. syst.

exits Edit.